Dynamic Fault Diagnosis in Mobile Ad Hoc Networks

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

(Specialization: Computer Science)

by

Madhu Chouhan

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769 008, India

May 2011

Dynamic Fault Diagnosis in Mobile Ad Hoc Networks

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

(Specialization: Computer Science)

by

Madhu Chouhan

(Roll- 209CS1067)

Under the guidance of

Prof. Manmath Narayan Sahoo

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769 008, India

May 2011

Dedicated to My Parents

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

Certificate

This is to certify that the work in the thesis entitled ***Dynamic Fault Diagnosis in Mobile Ad Hoc Networks*** submitted by ***Madhu Chouhan*** is a record of an original research work carried out by her under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Science and Engineering with the specialization of Computer Science in the department of Computer Science and Engineering, National Institute of Technology Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Place: NIT Rourkela
Date: 30 May 2011

Prof. Manmath Narayan Sahoo
Assistant Professor
Dept. of Computer Science and Engineering
National Institute of Technology, Rourkela
Odisha-769 008

Acknowledgment

I am grateful to numerous local and global peers who have contributed towards shaping this thesis. At the outset, I would like to express my sincere thanks to Prof. Manmath Narayan Sahoo for his advice during my thesis work. As my supervisor, he has constantly encouraged me to remain focused on achieving my goal. His observations and comments helped me to establish the overall direction of the research and to move forward with investigation in depth. He has helped me greatly and been a source of knowledge.

I am very much obliged to Prof. P. M. Khilar for his guideline, advice and support during my thesis work.

I am very much indebted to Prof. Ashok Kumar Turuk, Head-CSE, for his continuous encouragement and support. He is always ready to help with a smile.I am also thankful to all the professors of the department for their support.

I am really thankful to my all friends. My sincere thanks to everyone who has provided me with kind words, a welcome ear, new ideas, useful criticism, or their invaluable time, I am truly indebted.

I must acknowledge the academic resources that I have got from NIT Rourkela.I would like to thank administrative and technical staff members of the Department who have been kind enough to advise and help in their respective roles.

Last, but not the least, I would like to dedicate this thesis to my family, for their love, patience, and understanding.

Madhu Chouhan

Abstract

Fault diagnosis in Mobile Ad-hoc Networks (MANETs) is very challenging task. Diagnosis algorithm should be efficient enough to find the status (either faulty or fault free) of each mobile in the network. The models in the literature are either for static fault or dynamic fault. Dynamic fault identification is more complex and difficult than static fault. In this thesis, we proposed Dynamic Distributed Diagnosis Model to identify dynamic faults arising during the testing phase of the diagnosis session. The model assumes that each node has fixed and same set of neighbours i.e. the MANET topology is static throughout the diagnosis session. Our model works on a network with n number of nodes, which is σ-diagnosable. Where σ is one less than the minimum degree of a node in the network. It has two variation based on dissemination method, first is simple flooding approach and second is based on spanning tree. The flooding based model consists of two phases; a testing phase and a dissemination phase. The spanning tree based model has three phase; a testing phase, a building phase and a dissemination phase. In testing phase, we have used the concept of heartbeat, where every mobile broadcasts a response message at fixed interval, so that a node can correctly be diagnosed by at least one fault free neighbour. Building phase constructs a spanning tree with fault-free mobiles. Dissemination phase, with the help of spanning tree, disseminates the local diagnostic views through the fault-free mobiles. After aggregating the entire views, initiator node disseminates the global diagnostic view to the fault free mobiles down the spanning tree. In this way, all fault free units reach to an agreement about the status of other nodes in the network. Further, we have given the proof of correctness and completeness of our model and found the time complexity, and compared the simulation results with the existing fault diagnosis protocols.

Contents

List of Figures

List of Tables

List of Abbreviations

DSDP	Distributed Self-Diagnosis Protocol
DDD	Dynamic Distributed Diagnosis
CBR	Constant Bit Rate
DSDV	Destination Sequenced Distance-Vector
DSR	Dynamic Source Routing
FSR	Fisheye State Routing
GSR	Global State Routing
MANET	Mobile Ad hoc Network
NS 2	Network Simulator version 2
OTcl	Object-oriented Tool Command Language
TCL	Tool Command Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WSN	Wireless Sensor Network

Chapter 1

Introduction

Introduction

Motivation

Objective

Organization of Thesis

Chapter 1

Introduction

1.1 Introduction

In Latin, '*ad hoc*' phrase means '*for this*', meaning '*for this special purpose only*', by expansion it is a special network for a particular application. An Ad-hoc wireless network consists of a set of mobile nodes (hosts) that are connected through the wireless links. In Ad-hoc wireless network, communication is based on the principle of broadcast radio channel and reception of electromagnetic waves. The varied characteristics of wireless networks as compared to their wired counter parts addresses various issues such as mobility of nodes, limited bandwidth, error prone broadcast channels, hidden and exposed terminal problems and power constraints [2].

An important problem in designing MANET is handling failure of nodes. Each node in the system can be in one of two states faulty or fault-free. The nodes may fail because of battery discharge, crash or limitation in age. In this thesis work, we consider the fault to be permanent i.e., a faulty node remains faulty until it is repaired or replaced. However we consider both hard faults as well as soft faults. Soft faulted units can communicate with its neighbors but with altered behaviors, where as hard faulted units cannot communicate with its neighbors at all. Again we consider both static and dynamic faults. Static faults cannot arise during diagnosis session but dynamic faults can. Fault identification is one of the important parts in many protocols. When any altered behavior is shown by system or nodes of the network, a diagnosis function is started to determine which node(s) has(have) shown abnormal behavior. This is termed as Diagnosis;

diagnosis is classified based on the occurrence of fault. It is simply classified as static diagnosis and dynamic diagnosis. In static diagnosis, the faults are not occurring during the diagnosis session. In dynamic diagnosis, the faults can occur during the diagnosis session, which is difficult to handle because node can be faulty after it has been diagnosed as fault-free by other node.

Many previous models have been introduced to diagnose the network; A comparison based distributed fault diagnosis protocol for ad-hoc network called Static DSDP [3], which used flooding method to disseminate the information, pushed the message complexity up. To reduce the message complexity of static-DSDP, Dynamic-DSDP [2] has been proposed by Elhadef et al., which assumes a σ-diagnosable scenario that means each node connected with at least one fault-free neighbor. Dynamic-DSDP used spanning tree based dissemination approach to reduce the message complexity with extra overhead of spanning tree building time. The above two models deal with the permanent fault. Subbiah et al. [4] introduced a model called HeartbeatForward for partially connected network, they introduced dynamic fault diagnosis model for hard fault. The advantage of this protocol are, introducing dynamic fault diagnosis as well as taking less time to complete diagnosis, but the disadvantage of this protocol is that it uses flooding method which causes high message complexity.

In this thesis, we proposed two models for dynamic fault diagnosis in MANET. We have used comparison based diagnosis method to detect faulty and faulty-free nodes. That means we successfully detect the dynamic faults during Testing Phase. For disseminating the correct information to other node we have used two approaches; in first approach called Dynamic Distributed Diagnosis with Flooding (DDD Flooding), flooding method has been used and in second approach called Dynamic Distributed Diagnosis with Spanning Tree (DDD Spanning Tree), spanning tree method has been used. The spanning tree based model has three phase; a testing phase, a building phase and a dissemination phase. In testing phase, we have used the concept of heartbeat, where every mobile broadcasts a response message at fixed interval, so that all nodes are correctly diagnosed by at least one fault free neighbour. Building phase constructs a spanning tree with fault-free mobiles. Dissemination phase, with the help of spanning tree, disseminates the local diagnostic views to the parents. After aggregating the entire views, initiator node disseminates the global diagnostic view to the fault free mobiles down the spanning tree. In this way, all fault free units reach to an agreement about the status of other nodes in the network.

1.2 Motivation

After the study, we found that the presence of faulty node affects the efficiency and throughput of the network, which makes the network inconsistent. Faulty nodes cannot communicate with the other mobiles or behave unexpectedly and send unexpected results. In this way it unnecessarly consumes energy and cause inconsistency. Many protocols introduced by researchers to identify the fault in ad-hoc network are for static diagnosis, where node cannot change their status during diagnosis session. The fault (hard or soft) identification in dynamic diagnosis is more complex than static diagnosis; during the diagnosis fault-free node can be faulty.

1.3 Objective

Our objectives are:

- To design and develop a distributed system level diagnosis algorithm for identifying the fault status of various nodes in mobile ad-hoc networks where nodes are subjected to hard and soft faults under a dynamic faults environment

- To analyze and validate the performance of the proposed distributed diagnosis algorithm using standard simulator NS-2(v2.34).

- To compare the proposed method with the existing algorithms based on message and time complexity.

1.4 Thesis Organization

The rest of the thesis is organized as follows: In Chapter 2, we discuss about the background concepts related to our work. In Chapter 3, we discuss about the literature surveys that have been done during the research work. In Chapter 4, we proposed Dynamic Distributed Diagnosis model with two variant; Flooding based and Spanning Tree based. Analysis of the proposed model has done in Chapter 5. In Chapter 6, we discuss about the simulator used, simulations and results of our proposed model and compare the results with existing models. Finally in Chapter 7, we conclude and give the scope for the future work.

Chapter 2

Background Concepts

Chapter 2
Background Concepts

2.1 Introduction

In wireless networks, transmission is done from node to node. Each node acts as a router for transmitting and receiving packets to/from other nodes. An ad-hoc network connection is temporarily created to transmit the data. If the network is established for a long time, it is called simple local area network (LAN).

A wireless network uses a decentralized base station to which all nodes must communicate with. A peer-to-peer connection can increase the distance of the wireless network.

The different types of networks presented are Wired and Wireless networks. Wired networks are different from wireless networks. Wired network is connected from point to point. These networks are usually more efficient, less cost and faster than wireless networks due to their strong linking with the help of Switches and Hubs. After establishing the connection there is less chance of disconnection. Speed of wired network is about 100bps to 1000bps.

Wireless networks use radio frequencies waves to transmit and receive data instead of using some physical cables. The varied characteristics of wireless networks as compared to their wired counterparts, address various issues such as mobility of nodes, limited bandwidth, error prone broadcast channels, hidden and exposed terminal problems and power constraints.

Routing in ad-hoc networks has been a challenging task. The main cause for this is the constant change in network topology due to high degree of node mobility. A number of protocols have been developed to remove this problem.

An ad hoc network dynamically forms a provisional network without using any existing network infrastructure. The characteristics of ad-hoc network routing protocol are:

1. Simple
2. Less storage space
3. Loop free
4. Short control message (Low overhead)
5. Less power consumption
6. Multiple disjoint routes
7. Fast rerouting mechanism

A number of routing protocols like Ad-hoc On-Demand Distance Vector Routing (AODV), Dynamic Source Routing (DSR), Temporally Ordered Routing Algorithm (TORA) and Destination-Sequenced Distance-Vector (DSDV) have been implemented.

2.2 Application of Ad-Hoc Networks

The area of wireless networking emerges from the combination of cellular technology, personal computing and the Internet through this we can access information and services electronically, regardless of their geographic position. We can access continuously changing information from anywhere, anytime due to the increasing interaction between computing and communication. Wireless networks have become popular in the computing industry. The applications of the ad-hoc network are vast and interested reader may refer [2].

Wireless ad-hoc network is without any fixed infrastructure. Nodes are free to move randomly and generate random topology. The neighbors change due to the random movement of nodes. Ad-hoc networks are more appropriate in situations where a fixed infrastructure is not possible. Some application are:

Milirary Application

Ad-hoc network is initially developed for military application. Rapid formation of network and survivability are key requirements in military battlefield. In mission purpose applications such as a military needs security than other commercial or personal uses in MANET. A military scenario requires higher security for both information and topology. In such circumstances, we may need to blueprint the functionalities:

1. All secrete information is highly desirable to protect for confidentiality and integrity.

2. Military applications need to require network topology secret and don't allow traffic analysis. Routing protocol designers should try hard to hide the network topology from unauthorized.

Wireless Personal Area Networks (WPANs)
Wireless personal area network is used to cover small area within about 10 meters with limited transmission power. The network is created among personal computer and mobile computing device such as telephones and personal digital assistants through the wireless connection. The technology for WPANs is under development.

Disaster and Rescue Operations
Disaster or earthquake destroys communication and information system network. Residents cannot be used the disaster area. Therefore, it is required to reconstruct the communication and network system which can be quickly done by ad-hoc network.

A mobile ad-hoc network is also used in Bluetooth technology, which is designed for a private area network via wireless link between devices, such as printers and personal computers. An ad-hoc network system also supports Wi-Fi protocol.

2.3 Obstacles in Wireless Communication

The obstacles in wireless medium are interference, path loss, multipath propagation, and limited frequency spectrum. Interference occurs due to other radio frequencies and obstruction like wall. Path loss or path attenuation occurs due to decrease in the power density of electromagnetic waves. Path loss identified as the ratio among the powers of the transmitted signal to the receiver signal.

It depends on a number of factors such as radio frequency and the nature of the terrain. Multi propagation is signals travel from source to destination if there is obstacles between paths which make the signal propagate in paths away from the direct line of sight due to reflections, refraction and diffraction and scattering. In ad-hoc network, node shares a common broadcast radio channel since the radio spectrum is limited and bandwidth available for communication is also limited.

2.4 Faults

A node becomes faulty because of battery discharge, crash and limitation in age. An important problem in designing hosts MANET is handling failure of nodes is the distributed self diagnosis problem. In distributed self-diagnosis system each mobile node is able to diagnose the status of all nodes and knows the correct status of other nodes in the network.

2.4.1 Types of Faults

Each node in the system can be in one of two states faulty or fault-free. Faults can be categorized based on their duration, how it behaves after failure and occurrence of fault during diagnosis session.

Based on the Duration

Based on duration faults can be of three types:

1. Transient fault: A transient fault can disappear without any visible event; it appears in a network for short time. The recovery of transient faults from system is addressed using repeated-round techniques. A probabilistic model used for the action of faulty periods, and a fault analysis is used to obtain the optimum retry period.

2. Intermittent fault: It is problematic type of transient fault; we can't predict its appearance and disappearance in the network. An intermittent fault is occurred by several factors, some may be effect randomly, which occur simultaneously. These factors can only be identified when malfunction is occurred. Intermittent faults are difficult to identify and repair.

3. Permanent fault: Once it appears in network it remains until it removed and repaired by some external administrator. Permanent faults are simpler to deal.

Based on the Behavior

Based on behavior faults can be of two types:

1. Soft Fault: Soft faulted units can communicate with its neighbors but with unexpected behaviors and always give undesirable response.

2. Hard fault: Hard faulted units cannot communicate with its neighbors. It neither sends nor receives any information from the network.

Based on the Occurrence

Based on occurrence faults can be of two types:

1. Static fault: All faulty nodes be faulty from the starting of diagnosis session. The fault-free node can't be faulty during diagnosis session.

2. Dynamic fault: Fault-free node may become faulty during diagnosis session. It is hard to diagnosis because any node may fail after it diagnosed fault-free by any fault-free node.

Other Faults

Another type of fault is Byzantine fault which fail the components of a system in arbitrary ways by processing requests incorrectly. It is of two types:

1. Omission failures: This type of failure doesn't response for a request, e.g., crash, failing to receive a request, or failing to send a response.

2. Commission failures: This type of failure may respond in any unpredictable way, e.g., processing a request incorrectly, corrupting local state, and/or sending an incorrect or inconsistent response to a request.

2.5 Fault Diagnosis

Fault identification is one of the important part in many protocols. When the actual behavior is deviated by system or nodes of the system, a diagnosis function started to determine which node performed abnormal behavior that is called diagnosis. Diagnosis is classified based on the occurrence of fault. It simply can be classified as static diagnosis and dynamic diagnosis.

In static diagnosis, the fault does not occur during the diagnosis session; they already appeared in the networks. In dynamic diagnosis, the faults can occur during the diagnosis session, it is difficult to handle because node can be faulty after it has been diagnosed as fault-free by other node. We considered the problem of dynamic failures of node and remove those nodes from the network. Previously all work has dealt with the static fault situation where node cannot be faulty during diagnosis period.

2.5.1 Methods of Fault Diagnosis

Several diagnosis methods have been adopted based either on invalidation models, such as the PMC model, or comparison models, broadcast comparison model and the generalized comparison model [2]. The comparison model is most promising approachin which a set of task is assigned to nodes and outcomes are compared with their neighbor's outcomes. Various generalized comparison approach have been used. In this approach the comparison is done by the nodes themselves. The generalized comparison outcomes can be summerized as follows. If the tester and the tested nodes are fault-free, the comparison outcome is 0. If at least one of the tested nodes is faulty and the tester node is fault-free comparison outcome is 1. If the tester node is faulty, the comparison result is unpredictable (0 or 1) [2].

2.6 Conclusion

In this chapter, an overview of ad-hoc network and its application has been provided. We explained the basics of fault and its types, which is categorized based on duration, behavior and occurrence of fault during diagnosis session. Then we briefly explained definition of fault diagnosis and its methods.

Chapter 3

Literature Survey

Introduction

Literature Survey

Conclusion

Chapter 3

Literature Survey

3.1 Introduction

In this chapter we briefly discuss the research conducted so far for fault detection and diagnosis.

3.2 Literature Survey

The first model proposed for system-level diagnosis was the PMC model in 1967 [5], this model is named after author's initials: Preparata, Metze and Chien. The assumption of model is that a fault-free unit performs tests and generates results reliably. The PMC model gave necessary conditions for t-diagnosability, means at most 't' number of faulty nodes can be diagnosed by the system. This model is also used as symmetric invalidation model; faulty units can generate any wrong result.
Later Hakimi et al. [6] and Amin in 1974 characterized the PMC model. Each node is tested by at least 't' nodes, and $N \geq 2t + 1$, no two units test each other, and they gave necessary and sufficient conditions for a system to be t-diagnosable.

Another early model for system-level diagnosis is the BGM model [7], also named after authors' initials: Barsi, Grandoni and Maestrini in 1976. This model is similar to the PMC model. Its basic assumptions are: each test is executed by a single unit; each unit has the capability of testing any other unit; no unit tests itself. It is based on asymmetric invalidation rule; faulty units always generate wrong result. Table 3.1 contains the test results of various models. Similar models are proposed by Malek in 1980 [8], and by Chwa and Hakimi in 1981 [9]. These models assume a central observer which collects and examines the result about diagnosis. The MM model [8] assumes that comparisons are executed by the units themselves, and only results are sent to the central observer if both the units are fault free.

One more model proposed by Maeng and Malek in 1981 [10], it is a variation of MM model, assumes that node performs comparisons for its neighbors, no need of central observer for comparison and only comparison results are sent to the central observer if both the units are fault free. This model is called MM* model. The tester-node and tested-node are given in the Table 3.1.

Table 3.1: The invalidation rule for PMC, BGM, MM and Chwa and Hakimi's model.

Tester Node	**Tested Node**	**PMC Model**	**BGM Model**	**MM Model**	**Chwa and Hakimi's Model**
Fault-Free	Fault-Free	0	0	0	0
Fault-Free	Faulty	1	1	1	1
Faulty	Fault-Free	X	X	1	1
Faulty	Faulty	X	1	1	X

Sengupta and Dahbura simplify the MM model in 1992 [11]. In this model the comparator is compared by one of units. They also characterized diagnosable systems under the MM model. Probabilistic comparison-based models were first introduced by Dahbura et al. [12] in 1987, in this method processors perform comparison in the system. This model assumes a fault node based on probability after diagnosing the system.

Blough et al. [13] introduced the Broadcast Comparison based model in 1999. This model was generated for a fully distributed system, performed the comparison based approach to diagnosis the system. This model based on reliable broadcast. In this model a task is assigned to the different pair of nodes, which perform the task and send their outputs to all nodes in the system. All fault-free nodes compare all results and diagnose the system.

Other comparison based models introduced by Albini et al. [14, 15] in 2001 and 2005, that was fully distributed system but not a reliable broadcast. Here fault-free nodes perform test and categorize the system nodes in sets.

New promising application has approached by Chessa and santi in 2001 [3], first time they introduced the distributed diagnosis comparison model based on one-to-many comparison in ad hoc network. They diagnosed Permanent (hard and soft faults) and occurrence of fault was static. The model assumed that network topology doesn't change during diagnosis session and it is σ-diagnosable MANET. It used the asymmetric comparison based invalidations. In this model, every node receives different task from neighbours and every node responses for the different task. This model has used flooding approach to disseminate the diagnosis information.

Radhakrishnan et al. [16] in 2003, presented the distributed algorithm that adopts to the topology by utilizing spanning trees in the regions where the topology is stable and restoring to an intelligent flooding like approach in highly dynamic network. It is based on hold and forward or shuttling mechanism.

In 2004, Subbiah et al. [4], introduced a dynamic failure problem. To address this problem bounded correctness defined, which is made up of three properties: *bounded diagnostic latency*, which ensures that information about state changes of nodes in the system reaches working nodes with a bounded delay; *bounded start-up time*, which guarantees that working nodes determine valid states for every other node in the system within bounded time after their recovery, and *accuracy*, which ensures that no spurious events are recorded by working nodes. A node sends heartbeat message to a subset of other nodes and then rely those messages. The assumption of this approach is, heartbeats are the basic mechanism for a node to notify other nodes that it is working.

Rangrajan et al. [17] presented, a distributed algorithm for detecting and diagnosing faulty processor in an arbitrary network. A fault free node responds correctly within a specified timeout to test and forward diagnosis information correctly. A faulty node gives undesirable response.

Other new applications have given by Adya et al. [18] in 2004 and Pradeep Bahl [18] in 2003, presented an architecture for detecting and diagnosing faults in IEEE 802.11 infrastructure wireless network. Ronald et al. [19] presented new system level diagnosis called Adaptive DSD, which minimize the network resource. For diagnosis, they have assumed "symmetric invalidation" fault model of system diagnosis.

M. Elhadef et al. in [20] discussed a diagnosis approach for static topology in MANETs. This model presented the problem of self diagnosis of wireless mobile ad-hoc network. They have also used the comparison model for diagnosis. They have assumed that network is σ-diagnosable; means a network can diagnose at most σ faults. This protocol identifies hard and soft fault. Each node received task from neighbors and response exact by $\sigma + 1$ neighbor. They have used a spanning tree based approach to disseminate the diagnosis information.

Again M. Elhadef et al. presented two more approaches [2, 21], on fault identification in mobile ad hoc network, there is no restriction on the mobility of nodes. They introduced two approaches Adaptive-DSDP, this protocol includes three phases: self maintaining, testing phase and dissemination phase and Dynamic-DSDP nodes send the task along the response packet. Any node receives the response message and computes the task. After matching with the received response; correctly identify the fault status of nodes.

In 2010, Duarte et al. [22] presented a survey on fault diagnosis model. They have shown that the several models for comparison-based diagnosis differ in term of assumption.

3.3 Conclusion

This chapter involves the literature surveys that have been done during the research work. We have discussed about the related work that has been proposed by many researchers. The research papers related to fault and fault diagnosis from 1967 to 2010 has been shown which discussed about different methods and algorithm to diagnose the fault in the system.

Chapter 4

Proposed Model

Chapter 4

Proposed Model

4.1 Introduction

We proposed Dynamic Distributed Diagnosis Model to identify dynamic faults arising during the testing phase of the diagnosis session. The model assumes that each node has fixed and same set of neighbours i.e. the MANET topology is static throughout the diagnosis session. Our model works on a network with n number of nodes, which is σ-diagnosable. It has two variation based on dissemination method, first is simple flooding approach and second is based on spanning tree. The flooding based model consists of two phases; a testing phase and a dissemination phase. The spanning tree based model has three phases; a testing phase, a building phase and a dissemination phase. In testing phase, we have used the concept of HeartBeat, where every mobile broadcasts a response message at fixed interval, so that all nodes are correctly diagnosed by at least one of their fault-free neighbour. Building phase constructs a spanning tree with fault-free mobiles. Dissemination phase, with the help of spanning tree, disseminates the local diagnostic views to the parents. After aggregating the entire views, initiator node disseminates the global diagnostic view to the fault free mobiles down the spanning tree. In this way, all fault free units reach to an agreement about the status of other nodes in the network. In this chapter we have provided a brief description about the proposed model and its working principle with the help of algorithm.

4.2 System Model

A system is composed by n number of nodes called mobiles; they communicate with each other via a packet radio network. It is distributed dynamic diagnosis. Network delivers messages reliably. Each time the receiver of the message can identify its sender. Every node has its own id. Every node has 'M' number of predefined tasks with the task *ids*. The topology of the network at time 't' can be described as a directed graph $G(t) = (V, L(t))$, where V is the set of nodes, denoting mobiles and $L(t)$ is the set of links at time 't'. Given any $(x, y) \epsilon V$, edge $(x, y) \epsilon L(t)$ if and only if y is in the transmitting range of x at time t. At a certain time (i.e. during diagnosis), all the node has fixed number of neighbors i.e. at up to time t' i.e. $N_{t'}(x)$, where $(t \leq t' \leq t + T_{out})$. A channel access protocol is executed to solve the collision problems. The communication protocol supports a 1-hop reliable broadcast Primitive.

4.3 Fault Model

Faults are permanent. Nature of faults are *hard* and *soft* fault. Node can be faulty during the testing phase of diagnosis session. The fault free node can be faulty, but after send the last response message nodes are not allowed to change their status. Faulr-free node can be faulty but faulty node can not be fault-free. Maximum number of faulty node could be σ i.e. $\sigma \leq k - 1$ for connected network, where the k is minimum number of nodes if we remove k number of nodes then graph can disconnect. A diameter of graph is D_G. A MANET is called σ-diagnosable if all faulty mobiles can be unambiguously identified, provided the number of faulty mobiles doesn't exceed σ.

4.4 Diagnostic Model

Proposed model is a distributed diagnosis model. This model introduced a comparison-based diagnostic model for ad hoc networks. Proposed model considered the problem of fault-diagnosis during the diagnosis session of wireless and Mobile Ad hoc Networks (MANETs) using the comparison approach. In this approach, a network consists of a collection of n independent heterogeneous mobiles or stationary hosts interconnected via wireless links, they randomly pick any task from the memory and compute the task for the result and send to the neighbors. When the node receives the response then node compute the same task and matches the result. In dynamic diagnosis a fault-free node can be faulty during testing phase of the diagnosis session.

In the proposed model, every mobile will send the response periodically during the testing phase of the diagnosis session to identify the status of the mobile nodes.

4.5 Dynamic Distributed Diagnosis Model

In this thesis work we proposed Dynamic Distributed Diagnosis model, which is based on HeartBeatForward [4] dynamic fault identification model. HeartBeat-Forward model identified the faulty nodes for the partially connected network; the nature of the fault is *hard fault*. In the proposed model we find the *soft* as well as *hard faults* in mobile ad hoc network. Our model works under the assumptions described in the previous sections. The main idea is to identify the dynamic fault in the ad hoc network during the Testing Phase of the diagnosis session. Based on the dissemination method we have taken two variant of Dynamic Distributed Diagnosis model, first uses simple flooding approach (DDD Flooding) and second is based on spanning tree (DDD Spanning Tree). The flooding based model consists of two phases; Testing phase and Dissemination phase. In the Spanning tree base model diagnosis session is divided into three main phases: Testing Phase, Building phase and Dissemination phase.

4.5.1 Testing Phase

The main part of the fault diagnosis model is Testing Phase, which conclude for a mobile node that which of its neighbors are *faulty* and which are *fault-free*. At the end of this phase every node possesses list of faulty and fault free neighbors. The algorithm for Testing Phase is shown in Algorithm-1. First the data structure of mobile node x is listed. Node x receives following packets from the mobile node y during the Testing Phase:

Algorithm 1 Testing Phase

Data Structure for any mobile node x :
$N'_t(x)$: neighbors set of mobile node x at the time diagnosis i.e. $t \leq t' \leq t + T_{out}$.
$F(x)$: set of mobile nodes diagnosed as faulty, initialized to ϕ.
$FF(x)$: set of mobile nodes diagnosed as faulty-free, initialized to ϕ.
t_x : current time (associated with mobile node x).

```
mobile node x will recieve following packets from mobile node y ∈ N(x):

INIT_DIAGNOSIS : < INIT_DIAGNOSIS, θ, T_out >
if ((not)INIT_DIAGNOSIS_x) then
    INIT_DIAGNOSIS_x ← true;
    TIMEOUT_x ← t_x + T_out;
    rebroadcast the INIT_DIAGNOSIS packet;
    SENDLIFE_RESPONSE(θ);
else
    Drop the packet;
end if
LIFE_RESPONSE : < LIFE_RESPONSE, id_y, id_task, R_y, seq_no_y >
if ((not)TIMEOUT) then
    if ((< id_y, seq_no_y > ≠ < id_y, seq_no_last_seq_no >)and(id_y ∉ F_x)) then
        node[y].last_seq_no ← seq_no_y;
        generate the response R for task having task id i.e. id_task and compare with R_y;
        if (R_y == R) then
            FF_x ← FF_x ∪ {y};
        else
            F_x ← F_x ∪ {y};
            FF_x ← FF_x − {y};
        end if
    else
        Drop the packet;
    end if
else
    Drop the packet;
end if
TIMEOUT : when timeout occur to the mobile node x i.e. the Testing Phase of diagnosis session
is over.
TIMEOUT ← true;
F_x ← F_x ∪ {N_x(t') − (FF_x ∪ F_x)};
for all l ∈ FF_x do
    if (node[l].last_seq_no < node[x].last_seq_no) then
        F_x ← F_x ∪ {l};
        FF_x ← FF_x − {l};
    end if
end for
```

`INIT_DIAGNOSTIC`: We consider a fault free node as initiator, which will generate the `INIT_DIAGNOSTIC` packet to intimate the mobile nodes of the network about the starting of Diagnosis session and flood it to the network. The format of the packet is as follows: $<$ `INIT_DIAGNOSTIC`$, \theta, T_{out} >$, where θ is the time interval for generating the `LIFE_RESPONSE` packet and T_{out} is the time duration of the testing phase. Each node after receiving the `INIT_DIAGNOSTIC` packet first time, performs following things:

- Set $INIT\_DIAGNOSTIC_x$ as true.

Algorithm 2 SENDLIFE_RESPONSE

```
procedure SENDLIFE_RESPONSE(θ)                    ▷ where θ is the time intervel
    if (t_x == θ  and  t_x ≠ Timeout_x) then
        randomly pick the task from the memory and generate the response R_x for lask i;
        seq_no_x ← lastseq_no_x + 1;
        l_rb(LIFE_RESPONSE, id_x, id_task, R_x, seq_no_x);
        SENDLIFE_RESPONSE(t_x + θ);
    end if
end procedure
```

- Call the procedure $SENDLIFE\_RESPONSE$ with θ i.e. the time interval as the parameter.
- Set $Timeout_x$ as current time t_x plus T_{out} and rebroadcast the `INIT_DIAGNOSTIC` packet.

The procedure $SENDLIFE\_RESPONSE$ is used to generate the `LIFE_RESPONSE` packet at every θ interval till the timeout occurs. The procedure is defined more precisely in Algorithm-2.

`LIFE_RESPONSE`: After getting the intimation of the initialization of the diagnosis session the mobile node sends the `LIFE_RESPONSE` packet with the interval θ till timeout occurs. The format of the packet is $< \texttt{LIFE\_RESPONSE}, id_y, id_{task}, R_y, seq\_no_y >$, where id_y is the sender id, id_{task} is the task id, R_y is the response generated by node y with task id_{task} and sequence number of the response. The mobile node who receives the `LIFE_RESPONSE` packet does the following things:

- Check if TIMEOUT occurs, the receiver node does not process the `LIFE_RESPONSE` packet, otherwise it process the packet and check further.
- If mobile node previously received the `LIFE_RESPONSE` with same sequence number or the sender node is in the faulty set, then it drops the packet, otherwise does further processing.
- After receiving the fresh response packet by the node it updates the last sequence number received and generates the response of the task for given task id.
- Compares the generated response R with the received response R_y.

- If both responses are same, add the sender node *id* to the fault-free node set; otherwise add the sender node *id* to the faulty node set and remove the sender *id* from the *fault-free* node set.

In this way, the hard faults can be found after the timeout occurs to the mobile node.

`TIMEOUT`: When timeout occurs to the mobile node i.e. the time delay expires for the Testing Phase then variable `TIMEOUT` become true and the mobile calculates the hard faulty neighbors with the given conditions in the algorithm. The neighbor nodes which doesn't respond till the timeout were considered as hard faulty nodes and the nodes stop sending the response after some time were also considered as hard faulty nodes and add those faulty node *ids* to the faulty set.

After the testing phase we do not allow any new faults into the network. Just after the timeout the initiator node starts sending the `ST_MSG` to construct the spanning tree in the Building Phase.

Algorithm 3 Building Phase

Data Structure for any mobile node x :
$ParentSelected$: set to **true** once the mobile x sends its `ST_MSG` message, initialized to **false**.
$Children_x$: the set of mobile nodes that are considered as children of node x in the Spanning Tree.
FF_y : fault free set of node y.
F_y : faulty set of node y.

mobile node x will recieve following packets from mobile node $y \in N(x)$:
`ST_MSG` **:** $< \texttt{ST\_MSG}, y, z >$
if $(y \in FF_x)$ **then**
    if $(x == z)$ **then**
        $Children_x \leftarrow Children_x \cup \{y\}$;
    else if $(ParentSelected ==$ **false**$)$ **then**
        $Parent_x \leftarrow y$;
        $ParentSelected \leftarrow$ **true**;
        $l\_rb(\texttt{ST\_MSG}, x, y)$;
        $SetTimer(Tout)$;
    end if
else
    Drop the packet;
end if
`TIMEOUT` **:** when the delay Tout has expired.
if $(Children_x == \phi)$ **then**
    $l\_rb(\texttt{LOCAL\_DIAGNOSTIC}, F_x, FF_x)$;
end if

4.5.2 Building Phase

For disseminating the diagnosis information in the network we used existing approach, based on spanning tree [21]. There are two popular methods; flooding based and spanning tree based. Flooding based method is very easy but consume more energy because of redundancy and complexity. Flooding doesn't require building phase to disseminate the information at all. Whereas spanning tree method consumes less energy and consumes less message complexity. Therefore, in the proposed model we use spanning tree based approach to disseminate the local as well as global diagnosis information to the network. In Building Phase we construct the spanning tree, as explained in [20, 21]. Construction of ST (spanning tree) is done by the set of *fault-free* nodes. In the algorithm described in Algorithm-3 `ST_MSG` packet is used to construct the ST. The initiator node initiates the building phase by broadcasting the `ST_MSG` packet with two information; *sender id* and the *parent id* of the sender with the format as follows: $< \texttt{ST\_MSG}, y, z >$

If a mobile node does not have parent, the sender node becomes its parent, if the sender is not faulty. After finding the parent, mobile node set the variable *ParentSelected* as true, so that it can ignore further `ST_MSG` message it receives from the fault free nodes. After that it broadcasts the `ST_MSG` as sender itself and with the determined parent node id, for making the children and intimating the parent node. After a mobile sends the `ST_MSG` message, another timer is set to the time bound T_{out}. If the parent node of the sender is itself then add the sender node id to the set of children. If timeout occurs, the node which does not have any children starts sending the local diagnosis information to its parents and initiates the dissemination phase.

4.5.3 Dissemination Phase

After the spanning tree has been constructed, all the leave nodes of the spanning tree start sending their local diagnostic information to their parents. After receiving the local diagnostic information of all its children, a parent will forward

Algorithm 4 **Dissemination Phase**
Data Structure for any mobile node x :
$SystemDiagnosed$: set to **true** once the states of all mobiles are identified, initialized to **false**.
$children$: initialized to ϕ.

mobile node x will recieve following packets from mobile node $y \in N(x)$:

```
repeat
    LOCAL_DIAGNOSTIC : < LOCAL_DIAGNOSTIC, F_y, FF_y >
    if (y ∈ Children_x) then
        FF_x ← FF_x ∪ FF_y;
        F_x ← F_x ∪ F_y;
        children ← children ∪ {y};
        if (Children_x == children) then        ▷ mobile x waits for all its children's diagnosis views.
            l_rb(LOCAL_DIAGNOSTIC, F_x, FF_x);
        end if
    end if
    GLOBAL_DIAGNOSTIC : < GLOBAL_DIAGNOSTIC, F, FF >
    if (x == initiator) then
        l_rb(GLOBAL_DIAGNOSTIC, F_x, FF_x);
        SystemDiagnosed ← true;
    end if
    if (y == Parent_x) then
        FF_x ← FF;
        F_x ← F;
        if (Children_x ≠ φ) then
            l_rb(GLOBAL_DIAGNOSTIC, F, FF);
        end if
        SystemDiagnosed ← true;
    end if
until (SyetemDiagnosed == true)
```

the aggregated local diagnostic information to its parent, by adding its own local diagnostic information. This process continues until all the local diagnostic information has reached the initiator node which is the root of the ST. Now Initiator node has the global diagnostic information of the fault status of the network and will forward it down the ST, which result in all *fault-free* mobile nodes having a global view of the network [21].

4.6 Conclusion

We have proposed an algorithm to detect and diagnose the faulty node in MANET. The described model has two variation depends on dissemination of local view to the network. Testing phase is same in both models. In DDD Flooding, after the testing phase every node of the network floods their local view and in the DDD Spanning Tree, after the testing phase fault-free nodes create one spanning tree and disseminate the local and global view through the spanning tree. Algorithm for each and every phases of our proposed model is described in this chapter.

Chapter 5

Proposed Model Analysis

Chapter 5

Proposed Model Analysis

5.1 Introduction

In the previous chapter we described the functionality of our proposed model by providing assumptions and algorithms. In this chapter we give the analysis on proposed model and provide the correctness proof, completeness proof, communication and Time complexity.

5.2 Correctness Proof

Fault-free mobiles diagnose and disseminate information correctly. If the status of any node is correctly identified by atleast one *fault-free* neighbor node at the end of testing phase then it is called correct partial local diagnosis and it is completely diagnosed at end of dissemination phase if every *fault-free* node has the correct status of all mobiles in the system, then correct dissemination is achieved.

Lemma 1 (**Partial Diagnosis**) *The σ-diagnosable MANET is modeled as the connected graph G = (V, L), let $x \in V$ and $N(x)$ indicates x's neighbors. If node x is fault-free, then node is correctly identify by atleast one fault-free neighbor node. Every node x has at least one fault free neighbor.*

Proof. Let assume that $\mid N(x) \mid \leq \sigma$, all are faulty. If we discard all neighbor of x, will generate the disconnected graph, and hence $\mid N(x) \mid \geq k$. According to this it will be $\sigma \geq k$. The assumption of our model is the total number of faulty nodes σ, should not exceed k, that means σ should less or equal to the number of neighbors i.e. $\sigma \leq k - 1$.

Lemma 2 (**Fault-Free Spanning Tree**) *Let $G = (V, L)$ be the connected graph which is σ-diagnosable and every node contains two sets FF and F' where F' denote the set of faulty mobiles such that $\mid F' \mid \leq \sigma$. In tree all fault-free node disseminates information not only to its neighbours also to all nodes in the network.*

Proof. Given that the graph G is connected and the number of *faulty* nodes $\mid F' \mid \leq \sigma < k$, by removing *faulty* nodes we get another connected graph G' by node set $V - F'$. Since $\mid F' \mid < k$ then every *fault-free* node must be connected by altleast one *fault-free* neighbor. In this way tree propagates correct information to all *fault-free* nodes.

Lemma 3 (**Correct Dissemination**) *Let $G = (V, L)$ be the graph which represents a σ-diagnosable MANET, and F' be the set of fault nodes in the network, and $\mid F' \mid \leq \sigma$. ST is constructed by the fault-free nodes and is used to disseminate all global information in the network by the root node in a finite time.*

Proof. Firstly we have to prove that status of each node is correctly diagnosed by at least one *fault-free* neighbor, then after we have to prove that spanning tree is constructed by *fault-free* nodes only. These two are already proved in Lemma 1 and Lemma 2 respectively. Each *fault-free* node participates to disseminate the correct information. In this way, correct dissemination is acheived by *fault-free* nodes.

5.3 Completeness Proof

Theorem 1 *Let $G = (V, L)$ be the graph which represents a σ-diagnosable MANET. At the end of the dissemination phase each fault-free node x knows the faulty node set F_x which is same as total faulty nodes in the network $F_x = F'$ and fault-free nodes FF_x which is $FF_x = V - F'$.*

Proof. To prove this theorem, firstly we have to prove that every *fault-free* node has correctly diagnosed the state of all its neighbors at the end of the diagnosis session and we have to prove that all leaf nodes have transmitted its own neighbor information to its parent and parent combines all its children information and disseminates in the spanning tree. These are already proved in Lemma 3. After receiving local information, the root node combines all local information and generates the global information, which is disseminated to each node in the spanning tree in finite time. Hence each *fault-free* mobile knows the correct status of every node in the network.

5.4 Message Complexity

Let n is total number of mobiles in the σ-diagnosable MANET. Different type of messages transmitted in diagnosis session are presented, with the message complexity for the proposed model DDD Flooding and DDD Spanning Tree respectively in the Table 5.1 and Table 5.2.

Table 5.1: The message complexity of proposed model.

Message Type	**Complexity of the message**	**Description of the message**
INIT_DIAGNOSIS	n	All nodes generate at most one message. One initiator node generates this packet and sends to the neighbor then neighbor broadcast this packet to its own neighbor.
LIFE_RESPONSE	βn, where β is (T_{out}/θ)	Each mobile generates LIFE_RESPONSE message. It depends on the no. of interval during testing phase.
FLOODING	n^2	Each node floods its own local view as well as others local view in the network.

Theorem 2 *The message complexity of proposed model DDD Flooding is* $n+\beta n+n^2$.

Proof. The total number of messages transmitted during the diagnosis session of the proposed model is $n + \beta n + n^2$.

Theorem 3 *The message complexity of proposed model DDD Spanning Tree is* $4n - 2 + \beta n$.

Proof. The total number of messages transmit during the diagnosis session of the proposed model is $(4n - 2 + \beta n)$.

5.5 Time Complexity

The time complexity is expressed in terms of the following bounds:

Table 5.2: The message complexity of proposed model.

Message Type	Complexity of the message	Description of the message
INIT_DIAGNOSIS	n	All nodes generate at most one message. One initiator node generates this packet and sends to the neighbor then neighbor broadcast this packet to its own neighbor.
LIFE_RESPONSE	βn, where β is (T_{out}/θ)	Each mobile generates LIFE_RESPONSE message. It depends on the no. of interval during testing phase.
ST_MSG	n	The worst case scenario occurs when all mobiles are fault free, including initiator all mobile send one ST_MSG to its neighbor.
LOCAL_DIAGNOSTIC	$n-1$	Each mobile, excluding the initiator, sends one LOCAL_DIAGNOSTIC to its parent.
GLOBAL_DIAGNOSTIC	$n-1$	In the worst case scenario the depth of the tree is $n-1$. Hence, $n-1$ GLOBAL_DIAGNOSTIC message need to broadcast to complete diagnosis, down the tree.

- D_G : diameter of graph $G(V, L)$ that represents the MANET.
- D_{ST} : depth of the spanning tree.
- T_{INIT} : maximum time elapsed between sending the INIT_DIAGNOSIS message to the neighbor and receiving the first LIFE_RESPONSE packet from the neighbor.
- T_{GEN} : maximum time elapsed between the reception of the LIFE_RESPONSE message and computing the own task to evaluate the received response.
- T_{PROP} : maximum time to propagate a message in the network.
- T_F : maximum time elapsed between flooding the LOCAL_DIAGNOSIS messages by all nodes and receiving the last LOCAL_DIAGNOSIS message by the nodes in the network.
- T_{OUT} : time delay of diagnosis session.

Lemma 4 *The time complexity of the init diagnosis session is $D_G T_{INIT}$.*

Proof. The initiator node starts sending INIT_DIAGNOSIS message to its neighbors. A neighbor node receives the packet and broadcasts to its neighbors. So the last fault free node to receive INIT_DIAGNOSIS message and send the INIT_DIAGNOSIS will do so in at most $D_G T_{INIT}$ time bound .

Lemma 5 *The time complexity of the testing phase is $D_G T_{GEN}\beta + T_{OUT}$.*

Proof. The last *fault-free* node to receive a `LIFE_RESPONSE` message and compute the own task will take at most $D_G T_{GEN}$. Every fault free node should send the number of `LIFE_RESPONSE` message equal to the number of interval β, hence the time complexity will be $D_G T_{GEN}\beta$. Any *fault-free* node takes at most T_{out} time to diagnose at least one *fault-free* neighbor. Hence the time complexity of testing phase is at most $D_G T_{GEN}\beta + T_{OUT}$.

Lemma 6 *The time complexity of spanning tree construction is $D_{ST} T_{PROP} + T_{OUT}$.*

Proof. Spanning tree is constructed by fault-free nodes; initiator node starts to send `ST_MSG` and after receiving all node sends their `ST_MSG` to its neighbor. So the time taken by this phase is $D_{ST} T_{PROP}$. Every node sets a timer and after the Timeout, leaf node must start to send local information to the parent node. So the time complexity of building phase is $D_{ST} T_{PROP} + T_{OUT}$.

Lemma 7 *The time complexity of the dissemination phase of DDD Flooding is $D_G T_F$.*

Proof. All the nodes in the network floods `LOCAL_DIAGNOSIS` message, it takes at most $D_G T_F$.

Lemma 8 *The time complexity of the dissemination phase of DDD Spanning Tree is $2D_{ST} T_{PROP}$.*

Proof. Dissemination phase is started by the leaf nodes i.e., the nodes without children. Leaf nodes send local message to their parent; parent node includes its children information to its own information and wait for all children local information. In this way initiator node receives the local information from its children. This time initiator node knows the correct information about all the nodes in the network, for this it takes at most $D_{ST} T_{PROP}$ time. Now an initiator node has global message to send to its children and sets its SYSTEM_DIAGNOSED as true.

When intermediate node receives global message, they send it to their children in this way all leaf nodes receive the global message and this takes at most $D_{ST}T_{PROP}$ time. So the total time complexity of dissemination phase is $2D_{ST}T_{PROP}$.

Theorem 4 *The total time complexity of DDD Flooding is* $D_GT_{INIT}+\beta D_GT_{GEN}+D_GT_F+T_{OUT}$.

Proof. The proof of this theorem is trivial. Lemma 4-7 describe the time complexites of each of the phase of the proposed model and collectively the model takes at most $D_GT_{INIT}+\beta D_GT_{GEN}+D_GT_F+T_{OUT}$.

Theorem 5 *The total time complexity of DDD Spanning Tree is* $D_GT_{INIT}+\beta D_GT_{GEN}+3D_{ST}T_{PROP}+2T_{OUT}$.

Proof. The proof of this theorem is trivial. Lemma 4-6 and 8 describe the time complexites of each of the phase of the proposed model and collectively the model takes at most $D_GT_{INIT}+\beta D_GT_{GEN}+3D_{ST}T_{PROP}+2T_{OUT}$.

5.6 Conclusion

In this chapter we have provided a brief theoretical analysis of our proposed model. Correctness and completeness proof have been givento support our model and we found the message and time complexity of DDD Flooding and DDD Spanning Tree.

Chapter 6

Simulation and Results

Chapter 6

Simulation and Results

6.1 Introduction

In this chapter we discuss about the simulator, simulation parameters used in the experiment and result analysis. In order to find and compare the message complexity and diagnosis latency of proposed models with some existing models, an extensive set of experiments have been done. We have taken altered scenario of the faulty and fault-free nodes to find the efficiency of different models. Finally with the help of comparison graphs we have provided some analysis and results. The performance of our protocol is compared to that of Chessa and Santi's Static-DSDP [3], Elhadef et al.'s Dynamic DSDP [21] and Subbiah and Blough's HeartBeatForward [4] protocols.

6.2 Network Simulator 2 (NS-2)

NS-2 is a simple an event driven simulation tool which provides a platform to analyze the static and dynamic nature of the communication networks. It is a suitable and open source tool to simulate wired as well as wireless network protocols like TCP, UDP and routing protocols. NS-2 provides the command line functions to run and an animation to see the network, how packets and nodes are moving during simulation. The outcome of the simulation is in the form of trace file which shows the packet flow, type of packet, time of packet send and receive, sender and destination address etc. With the help of trace file we can plot the graph and analyze the performance of the algorithm [1].

NS-2 consists of two key languages: C++ and Object-oriented Tool Command

Language (OTcl) [1]:

- C++ defines the internal mechanism (i.e., a backend) of simulation objects.
- OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a front end).
- The C++ and the OTcl are linked together using TclCL.
- Mapped to a C++ object, variables in the OTcl domains sometimes referred to as handles.

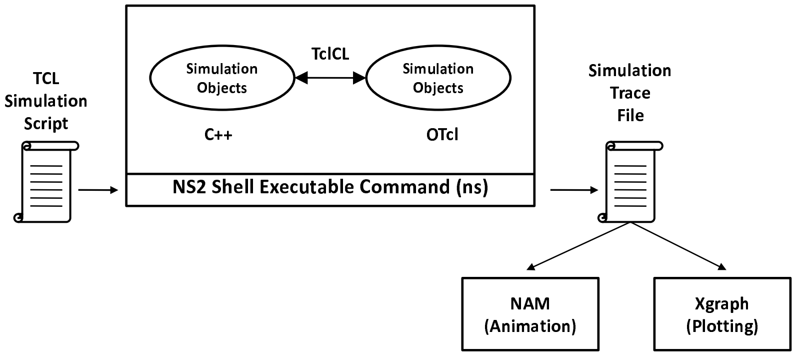

Figure 6.1: Basic Architecture of NS [1]

6.3 Simulation Parameters

To simulate our proposed models and existing models in NS-2 we used following parameters illustrated in Table 6.1.

Table 6.1: Simulation Parameters

S. No.	Simulation Parameters	Values
1	Simulator Used	Network Simulator (version 2.34)
2	Number of Nodes	10-100
3	Total number of Faulty nodes	1-10
4	Transmission range	200m
5	Area Size	1000m×1000m
6	MAC	802.11
7	Simulation Time	20Secs
8	Antenna model	OmniAntenna
9	Packet Size	512
10	Propagation Model	Two ray ground model
11	Speed	10m/s

6.4 Results

We have simulated five fault diagnosis models in NS-2. In which three are the existing models and two are the proposed models. The three existing models are: Static-DSDP [3], Dynamic-DSDP [21] and the HeartBeatForward [4]. The two proposed models are: DDD Flooding and DDD Spanning Tree. Here both Static-DSDP and Dynamic-DSDP are the models to diagnose the static faults and HeartBeatForward model diagnoses the dynamic hard faults. In our approach we allow dynamic hard as well as soft faults. Whereas the dissemination method of Static-DSDP and HardBeatForward used flooding method like our proposed model DDD Flooding and Dynamic-DSDP used spanning tree method like our proposed model DDD Spanning Tree.

Initially we have provided the comparison study of message complexity and time complexity between proposed and existing models, which is shown in Table 6.2.

Table 6.2: Comparison of Message and time complexity with existing models.

Models	Message Complexity	Time Complexity
DDD Spanning Tree	$4n - 2 + \beta n$	$D_G T_{INIT} + \beta D_G T_{GEN} + 3D_{ST} T_{PROP} + 2T_{OUT}$
DDD Flooding	$n + \beta n + n^2$	$D_G T_{INIT} + \beta D_G T_{GEN} + D_G T_F + T_{OUT}$
Static-DSDP	$n(1 + d_{max}) + n^2$	$(T_{GEN} + T_F) D_G + T_{OUT}$
Dynamic-DSDP	$n(1 + d_{min}) + 3n - 1$	$D_G T_{GEN} + 3D_{ST} T_{PROP} + 2T_{OUT}.$
HeartBeatForward	βn^2	$\beta T_{GEN} D_G T_F$

6.4.1 Efficiency with Respect to Number of Nodes

We analyzed DDD Spanning Tree approach with an extensive set of simulations and generated a graph regarding the result illustrated in Figure 6.2. We find that the message complexity of DDD Spanning Tree is linear for different values of θ. If the value of θ decreases value of β will increase, which causes a high message complexity. Here message complexity for $\theta = 1$ is more than that of $\theta = 2$ and 3.

The same set of tests we conduct to other protocols and obtained the results illustrated in Figure 6.3. We can see that message complexity of HeartBeatForward

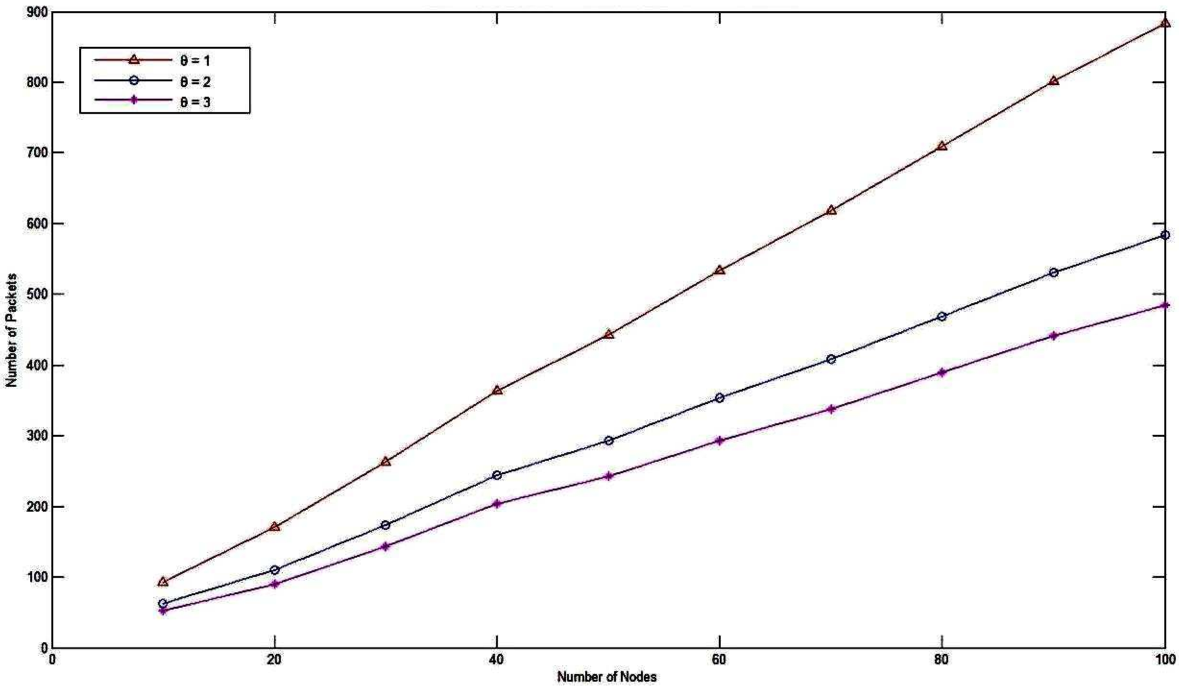

Figure 6.2: Message Complexity of DDD Spanning Tree with $\theta = 1, 2, 3$.

is of higher order than the other protocols. The reason behind this is the flooding technique used to send the HeartBeat to each node of the network. Chessa and Santi's Static DSDP is also showing the higher message complexity than proposed model DDD Flooding, because according to the algorithm in Static-DSDP every node responds to each to its neighbor. Likewise Elhadef's Dynamic DSDP responds to the minimum degree of the network.

Message complexity is the most significant factor of MANET. Every node's energy depends on the number of packet it sends or receives. So the energy consumed by a mobile is directly proportional to the amount of traffic it generates or receives [21].

The three consecutive graphs Figure 6.4, 6.5 and 6.6 show the message complexity of the HeartBeatForward, DDD Flooding and DDD Spanning Tree for different values of θ, and found that as HeartBeatForward used flooding method β times to diagnose a network, it produces a huge number of message in the network., whereas DDD Flooding used flooding once to disseminate the local information. As DDD Spanning Tree used spanning tree to disseminate the local information

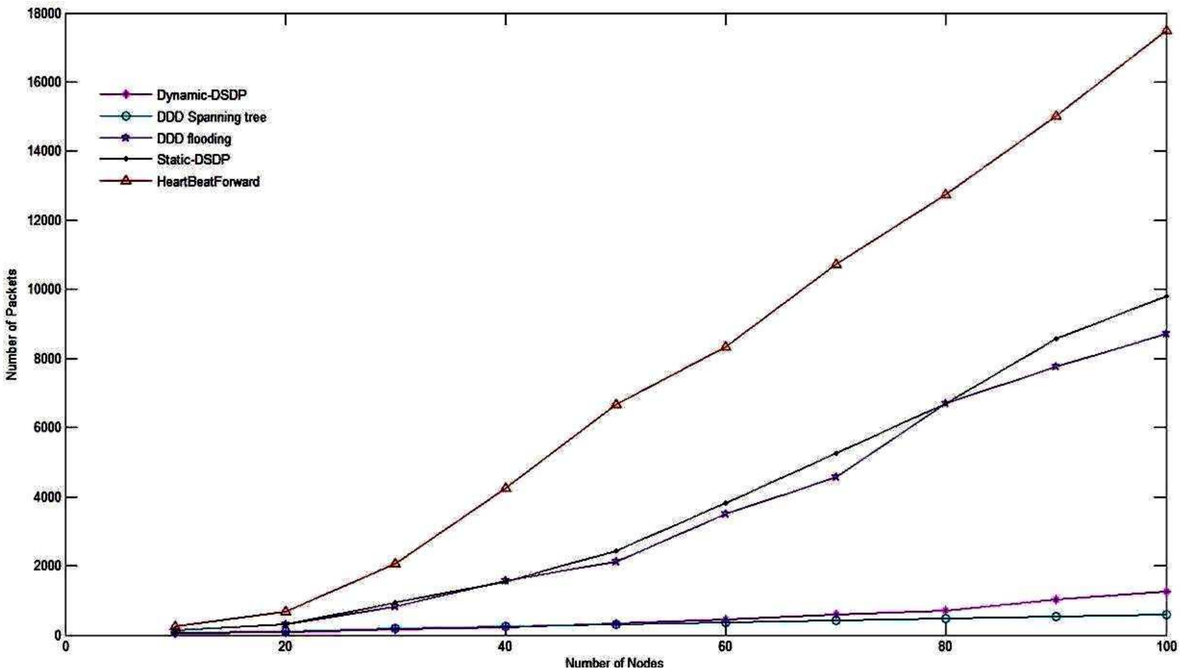

Figure 6.3: Comparison of Message Complexity with existing models

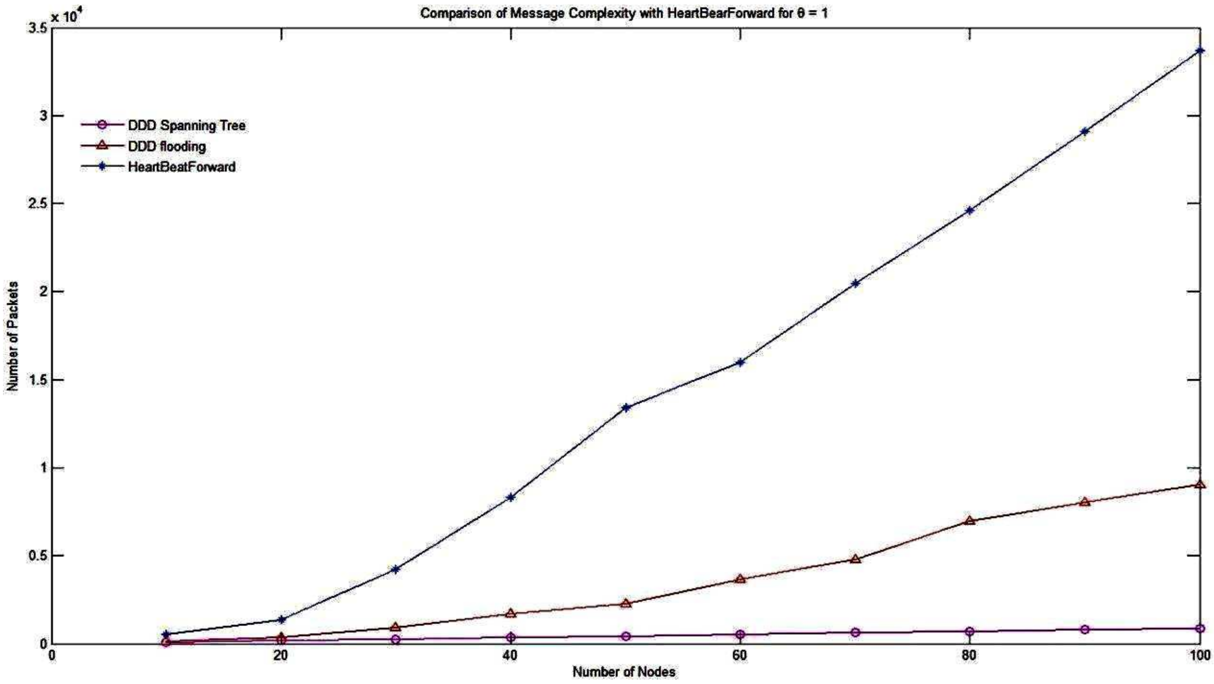

Figure 6.4: Comparison of Message Complexity with HeartBeatForward for $\theta = 1$

in the network which consume very less amount of messages.

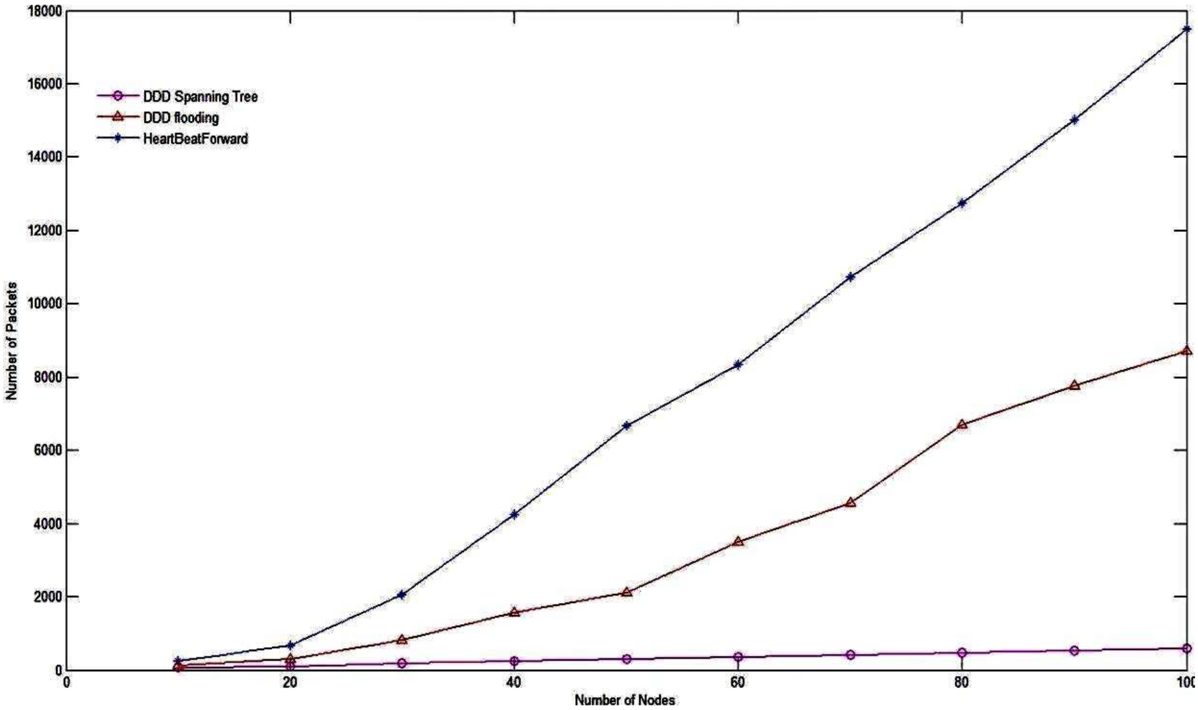

Figure 6.5: Comparison of Message Complexity with HeartBeatForward for $\theta = 2$

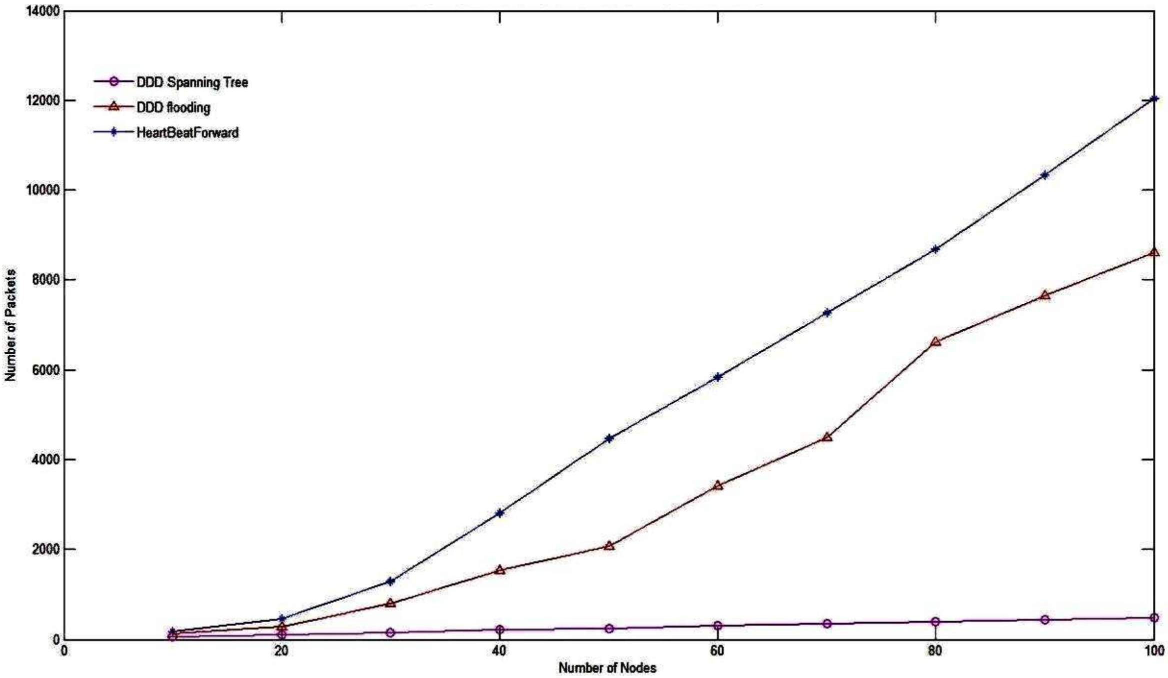

Figure 6.6: Comparison of Message Complexity with HeartBeatForward for $\theta = 3$

6.4.2 Efficiency with Respect to Number of Faults

For this simulation scenario we conduct experiment to find the message and time complexity. We have taken the same network with the minimal degree of 11, so that the number of faults can be at most 10. For the experiment we have taken fixed number of nodes in the network i.e. 100 and introduced the faults range from 1-10. Comparisons of message complexity with the existing models have been provided in the above graph in Figure 6.7. Here we increase the number of faults with the 100 nodes of network. To diagnose the number of faults with number of messages send is shown in the graph. We found that our proposed models DDD Flooding and DDD Spanning Tree outperform respectively as compared to Static-DSDP and Dynamic-DSDP.

In the DDD Spanning tree the message complexity will not vary in the testing phase and at the time of local dissemination. The algorithm restricts the faulty node to send message at the time of global dissemination, which takes very less message complexity, that is why the DDD Spanning Tree and Dynamic-DSDP has a horizontal line for the message complexity.

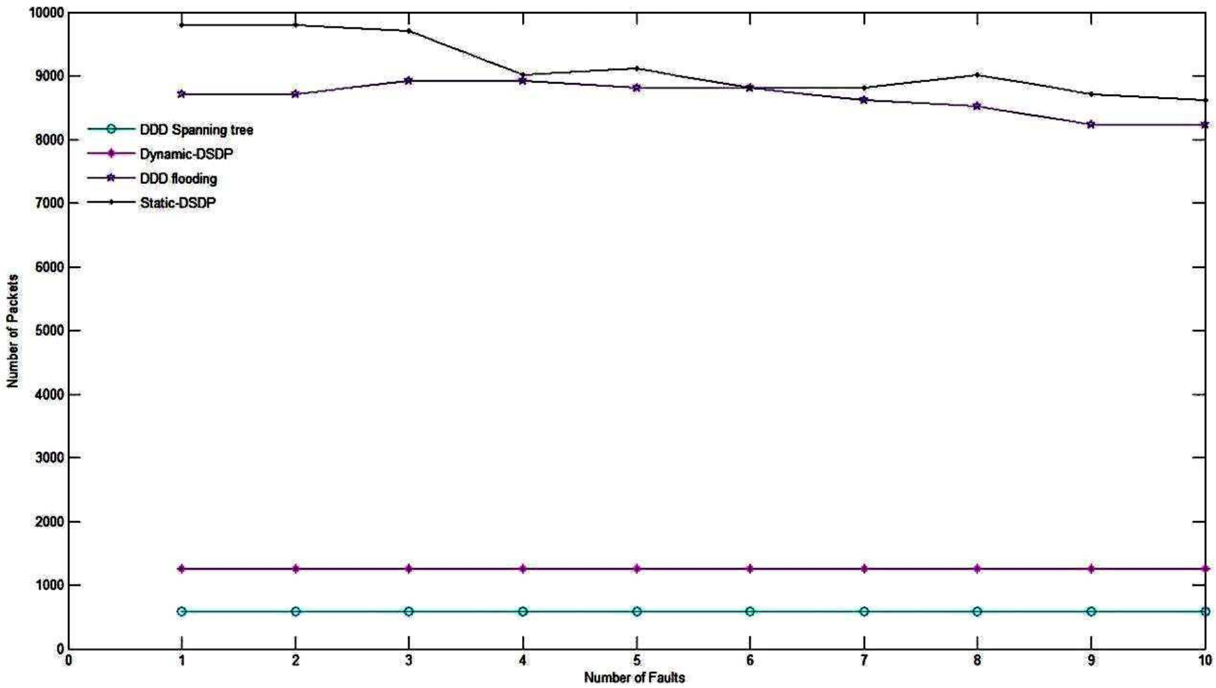

Figure 6.7: Comparison of Message Complexity with existing models

Figure 6.8 shows the comparison of message complexity with the HeartBeatForward. We observed that both the proposed model work efficiently in terms of message complexity while compared with the HeartBeatForward.

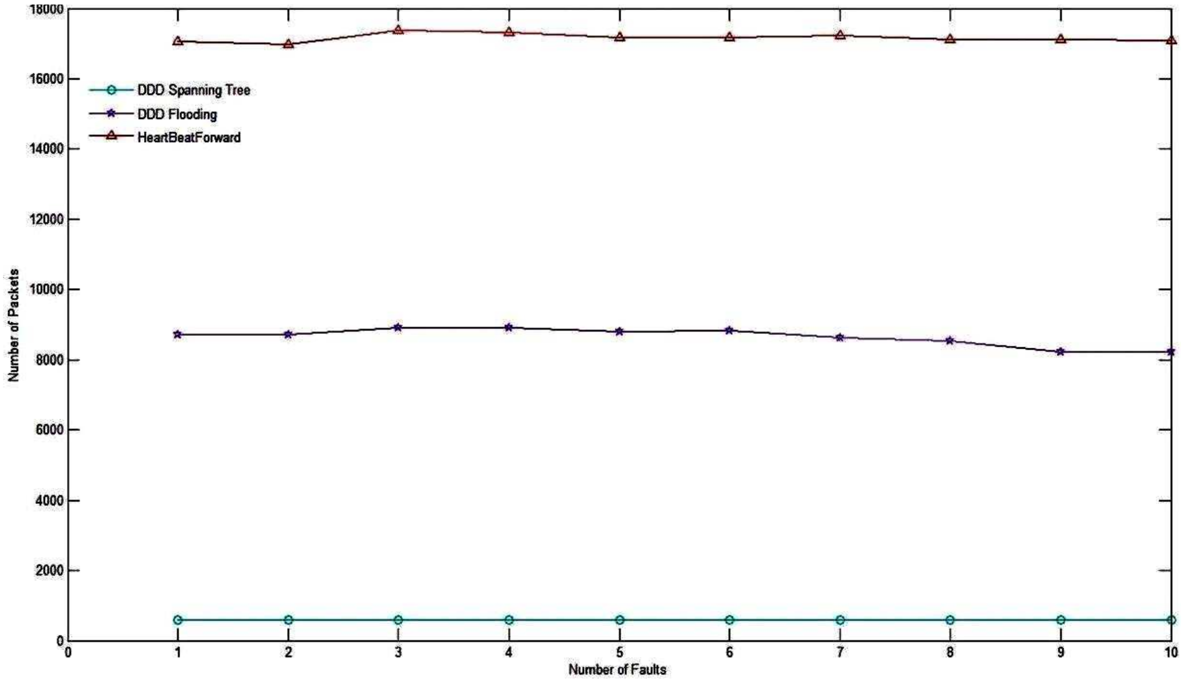

Figure 6.8: Comparison of Message Complexity with existing models

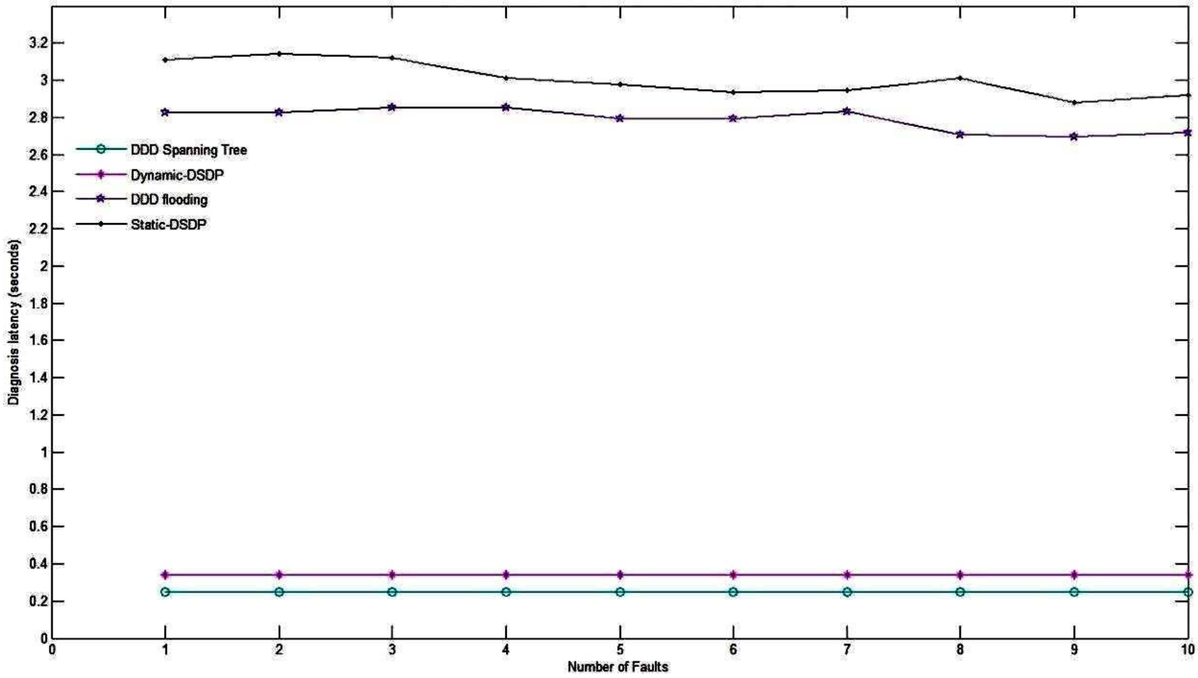

Figure 6.9: Comparison of Diagnosis latency with existing models

Figure 6.9 shows the comparison of Diagnosis latency with the existing models. Here also because the message complexity is high accordingly diagnosis latency is also high for the Static-DSDP. Compared to Dynamic-DSDP our model DDD Spanning Tree is more efficient in term of diagnosis latency.

6.5 Conclusion

In this chapter we discussed about the simulator and the simulation parameter we used for our experiments. We have also discussed the message and time complexity of other existing model. We have shown the simulation results in the form of graphs and brief discussions are provided. By the Simulation results we conclude that our approach diagnose the network more efficiently than the existing approaches.

Chapter 7

Conclusion and Future Work

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Diagnosis of dynamic faults is more complex than static faults. So far very less work has been done to identify dynamic faults. In this thesis, we proposed Dynamic Distributed model to diagnose dynamic hard and soft faults which occur during testing phase. Our model makes the fault-free node to correctly identify the status not only of its neighbors, but also the nodes of whole network. Our model is based on the HeartBeatForward and comparison approach in order to achieve a correct and complete diagnosis. In our model, once all mobiles have diagnosed the fault status of their neighbors, dissemination phase starts to spread the global information of all nodes to complete the diagnosis in network. Our model has two variation based on dissemination method; first is simple flooding based and second is based on the spanning tree that reduces the message complexity.

In this thesis we have provided the correctness and completeness proof of our algorithms. We also found the message and time complexity of the proposed models. Further we implemented our models along with some existing models in network simulator 2 (NS-2) and compared the results. After the simulation we analyzed the results and found that our approaches Dynamic Distributed Diagnosis with flooding (DDD Flooding) outperforms the Static-DSDP and HeardBeatForward and Dynamic Distributed Diagnosis with spanning tree (DDD Spanning Tree) is efficient than Dynamic-DSDP in terms of message complexity and diagnosis latency.

7.2 Scope of The Future Work

As our model is restricted to detect dynamic fault during the testing phase. In near future we will try to propose the algorithm which can identify the dynamic fault during the diagnosis session with less communication complexity.

Bibliography

[1] Teerawat Issariyakul and Ekram Hossain. *Introduction to Network Simulator NS2*. Springer Science and Business Media USA, LLC, 233 Spring Street, New York, NY 10013, USA, 2008.

[2] M. Elhadef, A. Boukerche, and H. Elkadiki. Diagnosing mobile ad hoc networks: two distributed comparison-based self-diagnosis protocols. In *In Proc. of the 4th ACM International Workshop on Mobility Management and Wireless Access Protocols*, October 2006.

[3] S. Chessa and P. Santi. Comparison-based system-level fault diagnosis in ad hoc networks. In *Proc. of the 20th Symp. On Reliable Distributed Systems*, pages 257–266, 2001.

[4] A. Subbiah and D. M. Blough. Distributed diagnosis in dynamic fault environments. *IEEE Transactions on Computers*, pages 453–467, 2004.

[5] F. Preparata, G. Metze, and R. T. Chien. On the connection assignment problem of diagnosable systems. *IEEE Transactions on Computers*, EC-16:848–854, 1967.

[6] S.L Hakimi and A. T. Amin. Characterization of connection assignment of diagnosable systems. *IEEE Transactions on Computers*, 23(1):86–88, January 1974.

[7] F. Barsi, F. Grandoni, and P.Maestrini. A theory of diagnosability of digital systems. *IEEE Transactions On Computers*, 25(6):585–593, June 1976.

[8] M. Malek. A comparison connection assignment for diagnosis of multiprocessor systems. In *Proc. of the 7th Annual Intl. Symp. on Computer Architecture*, pages 31–36, New York, NY, USA, 1980. ACM.

[9] K. Y. Chwa and S. L. Hakimi. Schemes for fault-tolerant computing: A comparison of modularly redundant and t-diagnosable systems. *Information and Control*, 49:212–238, 1981.

[10] J. Maeng and M. Malek. A comparison connection assignment for self-diagnosis of multiprocessor systems. In *Proc. of the 11th IEEE Fault-Tolerant Computing Symp.*, pages 173–175, 1981.

[11] A. Sengupta and A. T. Dahbura. On self-diagnosable multiprocessor systems: Diagnosis by comparison approach. *IEEE Transactions on Computers*, pages 1386–1396, 1992.

[12] A. T. Dahbura, K. K. Sabnani, and L. L. King. The comparison approach to multiprocessor fault diagnosis. *IEEE Transactions on Computers*, 36(3):373–378, March 1987.

[13] D. M. Blough and H. W. Brown. The broadcast comparison model for on-line fault diagnosis in multicomputer systems: Theory and implementation. *IEEE Transactions on Computers*, pages 470–493, 1999.

[14] L. C. P. Albini and E. P. Duarte Jr. Generalized distributed comparison-based system-level diagnosis. In *Proc. of the 2nd IEEE Latin American Test Workshop*, pages 285–290, 2001.

[15] R. P. Ziwich, E. P. Duarte Jr., , and L. C. P. Albini. Distributed integrity checking for systems with replicated data. In *Proc. of the 2005 11th IEEE International Conference on Parallel and Distributed Systems (ICPADS'05)*, pages 363–369, 2005.

[16] S. Radhakrishnan, G. Racherla, C. N. Sekharan, N. S. V. Rao, and S. G Batsell. Protocol for dynamic ad-hoc networks using distributed spanning trees. *Wireless Networks*, pages 673–686, 2003.

[17] S. Rangarajan, A. T. Dahbura, and E.A. Ziegler. A distributed system-level diagnosis algorithm for arbitrary network topologies. *IEEE Transactions on Computers*, pages 312–333, 1995.

[18] R. Chandra and P. Bahl. Multinet: Connecting to multiple ieee 802.11 networks using a single wireless card. In *Proc. of IEEE INFOCOM*, Hong Kong, March 2004.

[19] R.P. Bianchini Jr and R.W. Buskens. Implementation of on-line distributed system-level diagnosis theory. *IEEE Transactions on Computers*, 41(5):616–626, 1992.

[20] M. Elhadef, A. Boukerche, and H. Elkadiki. Performance analysis of a distributed comparison based self-diagnosis protocol for wireless ad-hoc networks. In *Proc. of 9th ACM Int. Symp. On Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 165–172, New York, NY, USA, 2006. ACM.

[21] M. Elhadef, A. Boukerche, and H. Elkadiki. A distributed fault identication protocol for wireless and mobile ad hoc networks. *Journal of parallel and distributed computing*, 68:321–335, 2008.

[22] E.P. Durte Jr., R.P. Ziwich, and L.C.P.Albini. A survey of comparison-based system-level diagnosis. *ACM Computing Surveys*, pages 1–65, 2010.

[23] J. Xu and L. Lilien. A survey of methods for system-level fault diagnosis. In *ACM-IEEE computer society fall joint computer conference*, pages 534–540, 1987.

[24] R.P. Bianchini Jr and R.W. Buskens. Implementation of on-line distributed system-level diagnosis theory. *IEEE Transactions on Computers*, 41(5):616–626, 1992.

[25] S. Rangarajan, A. T. Dahbura, and E.A. Ziegler. A distributed system-level diagnosis algorithm for arbitrary network topologies. *IEEE Transactions on Computers*, pages 312–333, 1995.

[26] R. Chandra and P. Bahl. Multinet: Connecting to multiple ieee 802.11 networks using a single wireless card. In *Proc. of IEEE INFOCOM*, Hong Kong, March 2004.

[27] A. Adya, P. Bahl, R. Chandra, and L. Qiu. Architecture and techniques for diagnosing faults in ieee 802.11 infrastructure networks. In *In Proc. of the 10th International Conference on Mobile Computing and Networking (MobiCom '04)*, pages 30–44, 2004.

[28] A. Adya, P. Bahl, R. Chandra, and L. Qiu. Architecture and techniques for diagnosing faults in ieee 802.11 infrastructure networks. In *In Proc. of the 10th International Conference on Mobile Computing and Networking (MobiCom '04)*, pages 30–44, 2004.

Dissemination of Work

Accepted

1. Madhu Chouhan, M. N. Sahoo and P. M. Khilar "Fault Diagnosis in MANET", *International Conference on Advance in Computing and Communications (ACC-2011)*,Kochi, Kerala, India, 22 - 24 September, 2011.